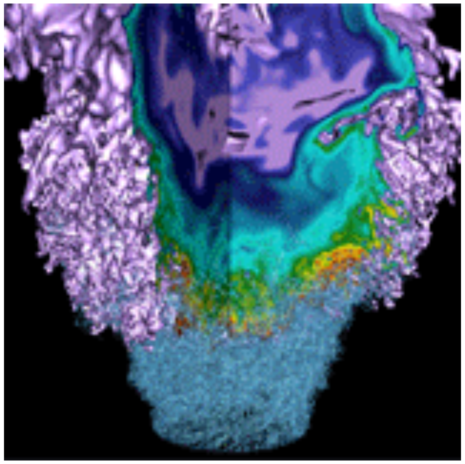
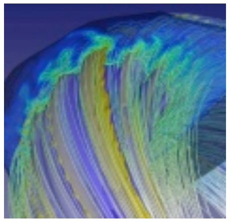
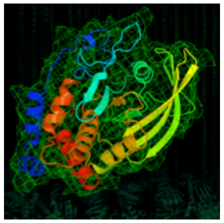
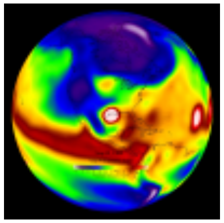
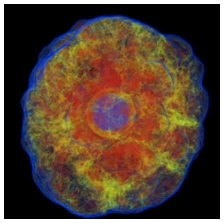
Scalable HDF5

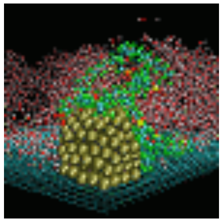












Quincey Koziol

ATPESC

July 31, 2020

koziol@lbl.gov

Why Use HDF5?

- **Have you ever asked yourself:**
  - How will I deal with one-file-per-process in the exascale era?
  - Do I need to be an “MPI and Lustre Pro” to do my research?
  - Why is my checkpoint taking so long?
- **HDF5 *hides I/O complexity* so you can *concentrate on your science***
  - Optimized I/O to single shared file*

* Prototypes of “multi-file” HDF5 storage are under development as well.

Goals

- **HDF5 Overview**
- **HDF5 Programming Overview**
- **Parallel HDF5**
- **Intro to Scalable HDF5**

WHAT IS HDF5?

What is HDF5?

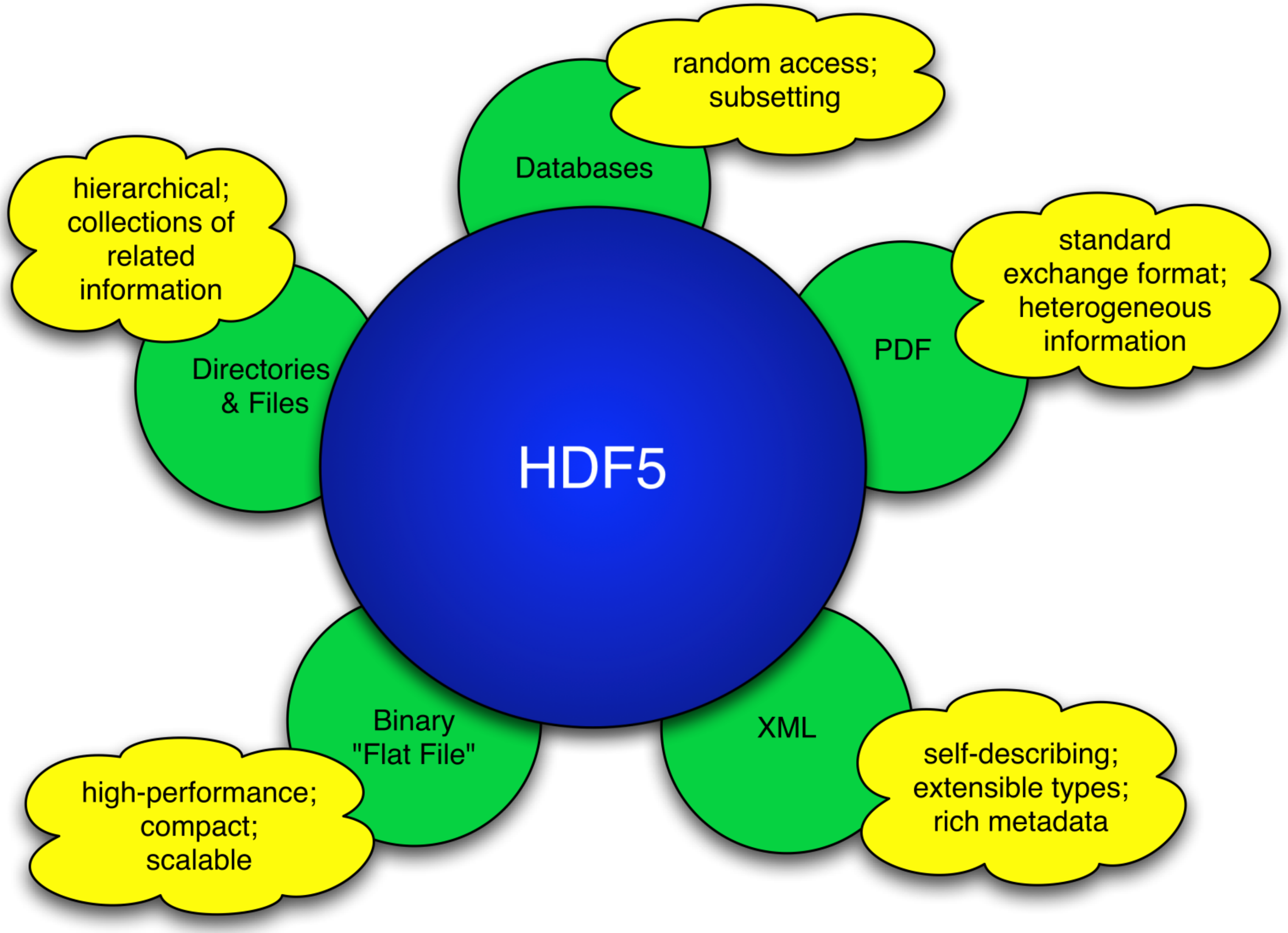
- **HDF5 == Hierarchical Data Format, v5**

- **Open file format**
  - Designed for high volume and complex data

- **Open source software**
  - Works with data in the format

- **An extensible data model**
  - Structures for data organization and specification

HDF5 is like …



HDF5

Databases

random access; subsetting

PDF

standard exchange format; heterogeneous information

XML

self-describing; extensible types; rich metadata

Binary "Flat File"

high-performance; compact; scalable

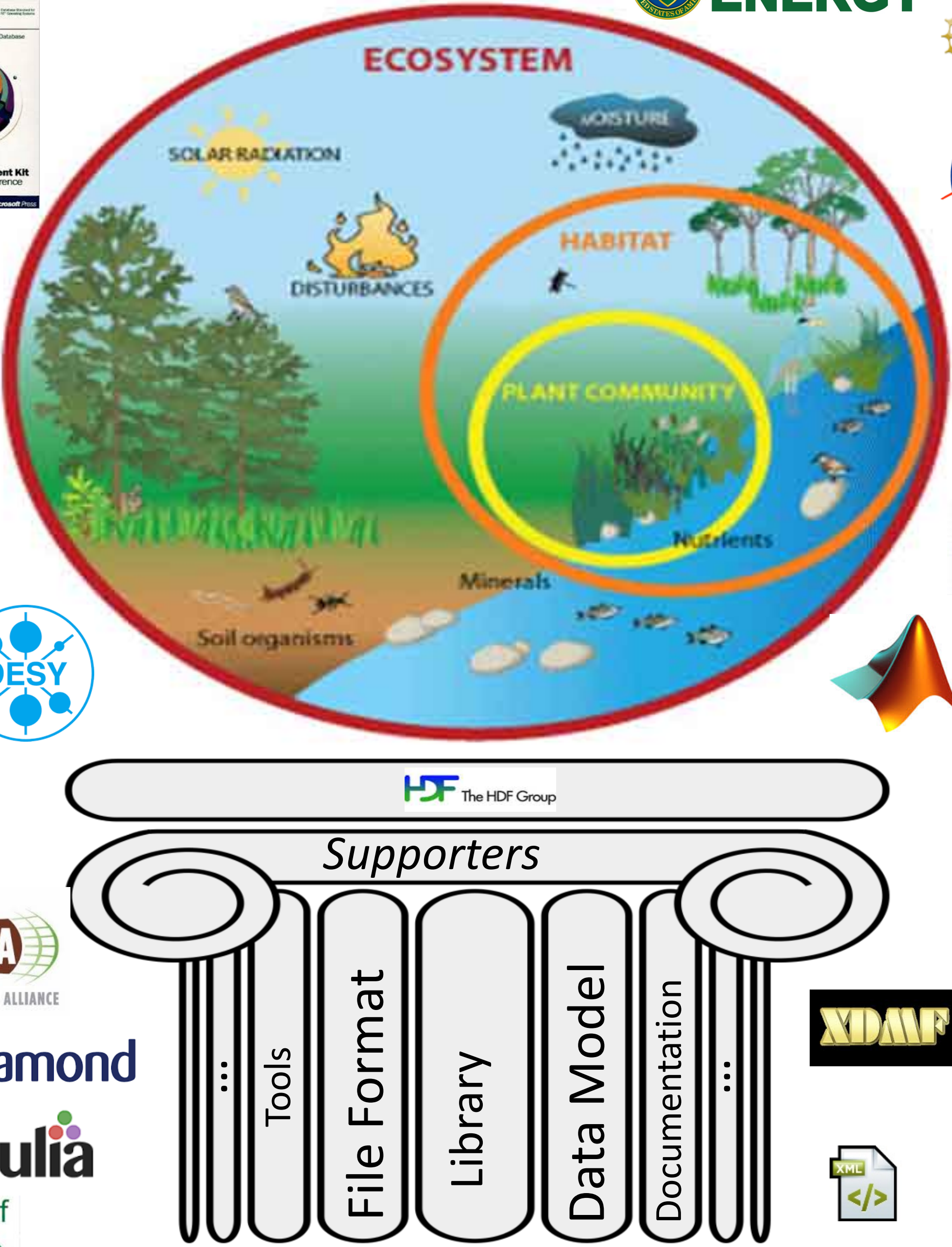
Directories & Files

hierarchical; collections of related information

HDF5 is designed …

- for high volume and/or complex data
- for every size and type of system (portable)
- for flexible, efficient storage and I/O
- to enable applications to evolve in their use of HDF5 and to accommodate new models
- to support long-term data preservation

HDF5 Ecosystem



Petrel

Microsoft ODBC 3.0 Software Development Kit and Programmer's Reference

Everything You Need to Build Easy Database Connectivity into Your Applications

Microsoft Press

CGNS cfd data standard

U.S. DEPARTMENT OF ENERGY

DESY

OPeNDAP

RaspberryPi

RDA RESEARCH DATA ALLIANCE

Atlassian

diamond

pandas

julia

U.S. DEPARTMENT OF ENERGY Office of Science

U.S. DEPARTMENT OF ENERGY

National Science Foundation WHERE DISCOVERIES BEGIN

NASA

HDF EOS

LIVE HDF5 HDF5 Toolkit for LabVIEW

NeXus

Biological Observation Matrix

MDTRAJ

netCDF

R

JSON

NOAA NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION U.S. DEPARTMENT OF COMMERCE

HDF5 for Lua

XDMF

PyTables

XML

O'REILLY

Python and HDF5

UNLOCKING SCIENTIFIC DATA

Andrew Collette

BERKELEY LAB Lawrence Berkeley National Laboratory

HDF5 DATA MODEL

HDF5 Data Model

HDF5 Objects

Dataset

Group

Attribute

Link

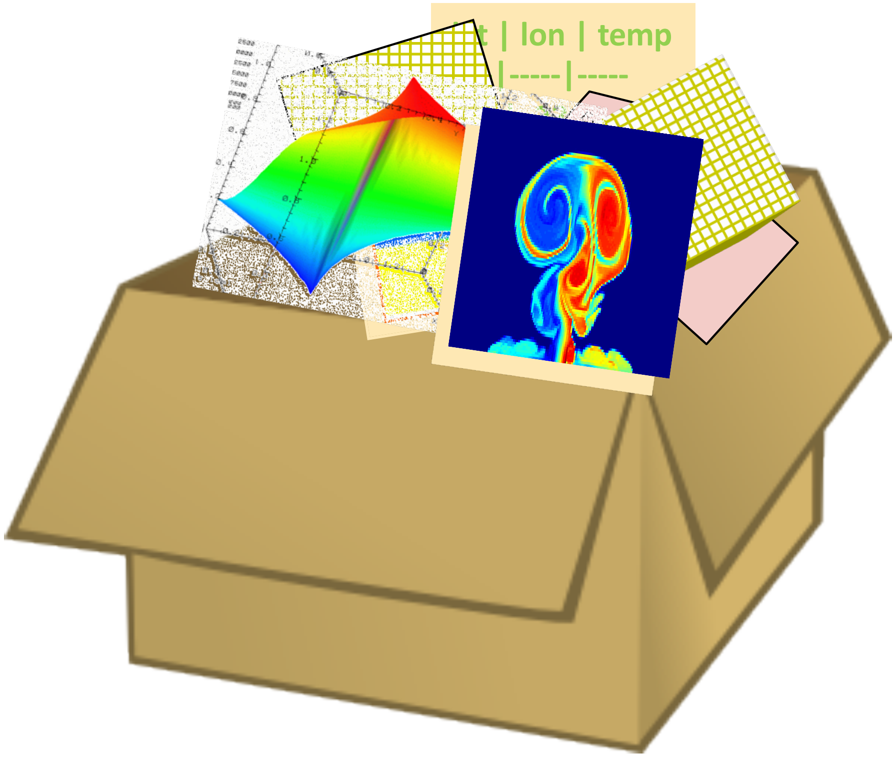
Datatype

Dataspace

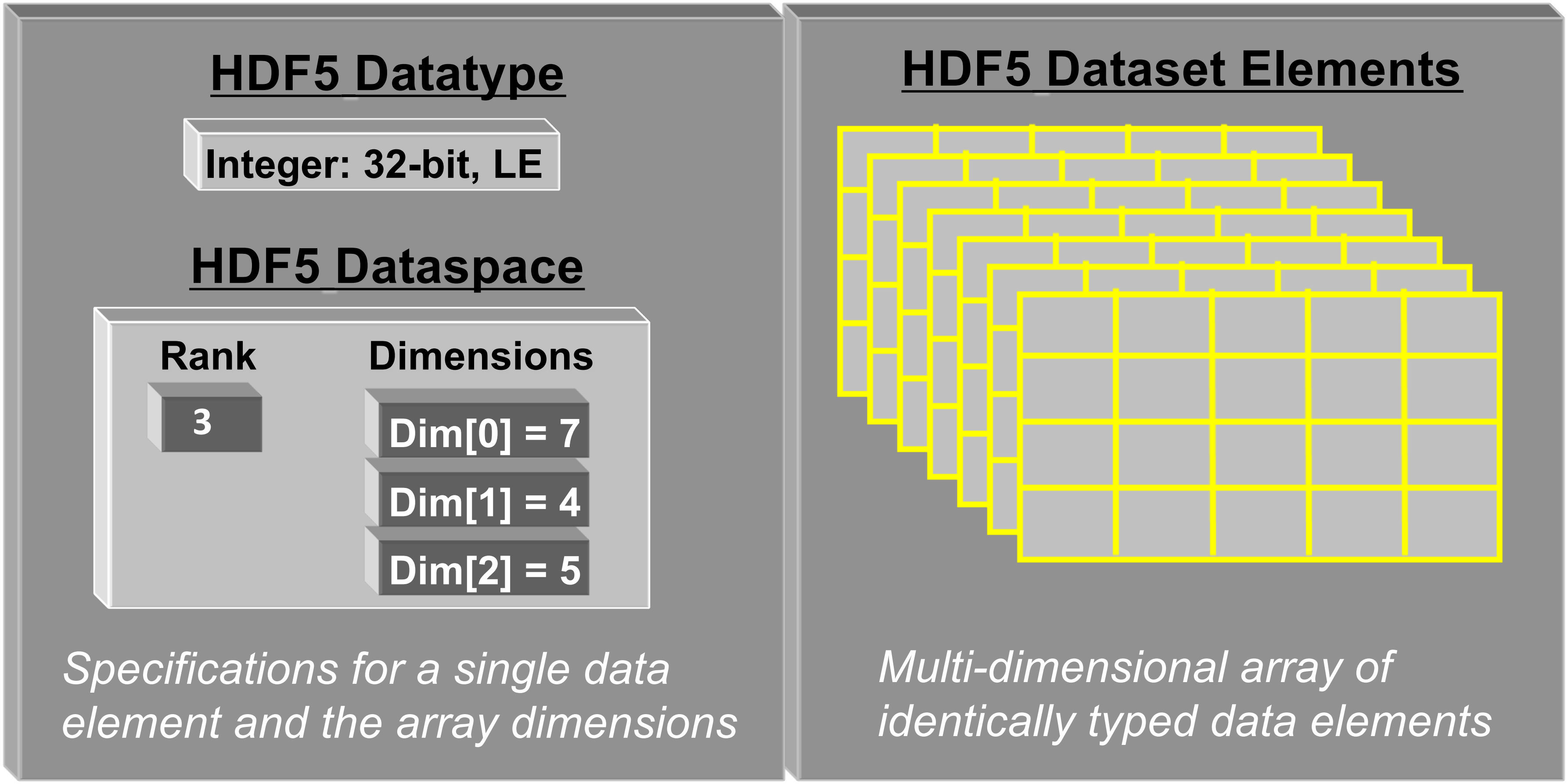
File

HDF5 File

An HDF5 file is a **container** that holds data objects.



HDF5 Dataset



HDF5 Datatype

Integer: 32-bit, LE

HDF5 Dataspace

Rank: 3

Dimensions: Dim[0] = 7, Dim[1] = 4, Dim[2] = 5

Specifications for a single data element and the array dimensions

HDF5 Dataset Elements

Multi-dimensional array of identically typed data elements

- HDF5 datasets **organize and contain** data elements.
  - HDF5 datatype describes individual data elements.
  - HDF5 dataspace describes the logical layout of the data elements.

HDF5 Dataspace

- **Describes the logical layout of the elements in an HDF5 dataset**
  - NULL
    - No elements (i.e., the empty / null set)
  - Scalar
    - Single element (a “point”, without dimensionality)
  - Simple (*most common*)
    - A rectangular array:
      - Rank = number of dimensions
      - Dimension sizes = number of elements in each dimension
      - Maximum number of elements in each dimension
        - » may be fixed or unlimited

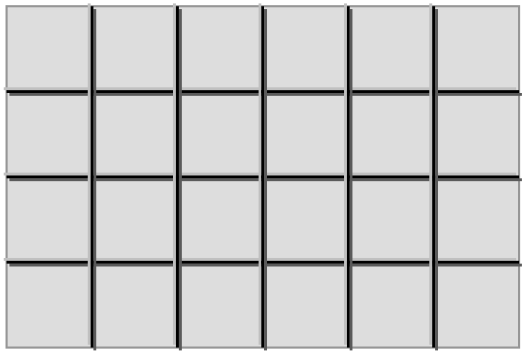
HDF5 Dataspace – Two Roles

Spatial information for Datasets and Attributes:

- Rank and Dimension sizes
- Permanent part of object definition

Rank = 2

Dimensions = 4 x 6

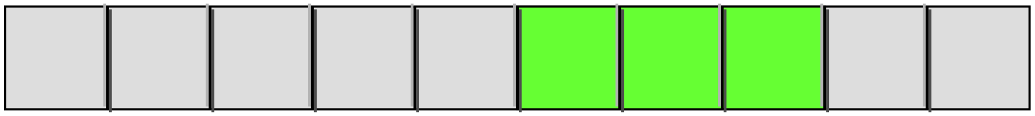


Partial I/O: Dataspace and selection describe application's data buffer and the elements participating in I/O

Dataspace:

Rank = 1

Dimension = 10



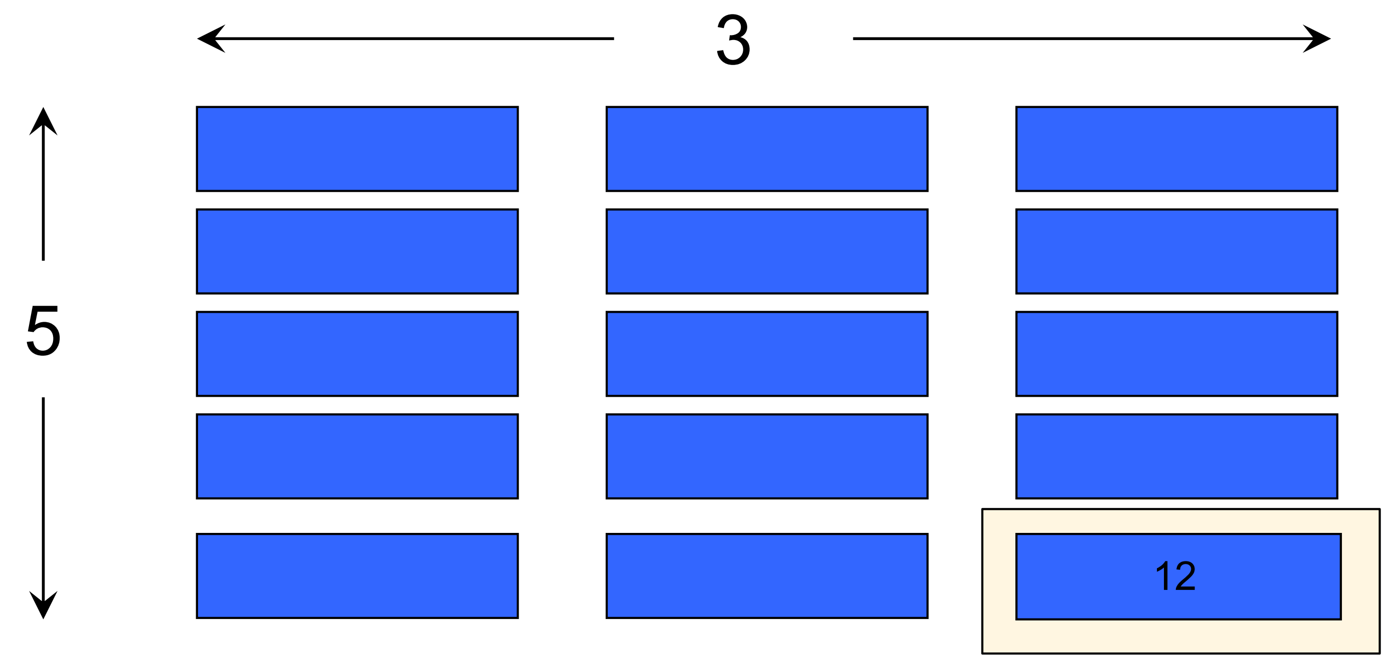
Selection: Start = 5

Count = 3

HDF5 Datatypes

- **Describe the individual data elements in an HDF5 dataset or attribute**
- Wide range of datatypes supported:
  - Integer
  - Float
  - Enum
  - Array (similar to matrix in math)
  - Variable-length sequence (e.g., strings, C++ vectors)
  - Compound (similar to C structs)
  - User-defined (e.g. 12-bit integer, 16-bit float, etc.)
  - More …

HDF5 Dataset

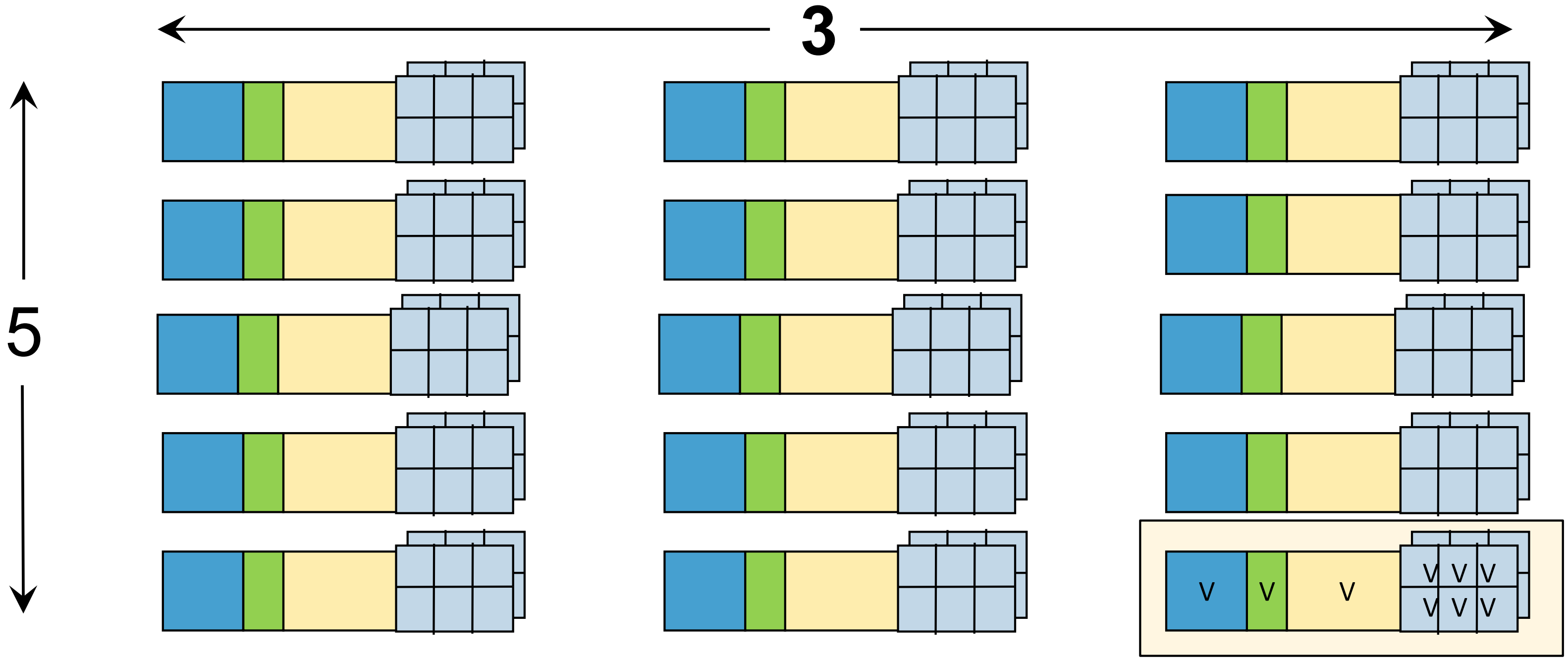


Datatype: 32-bit Integer

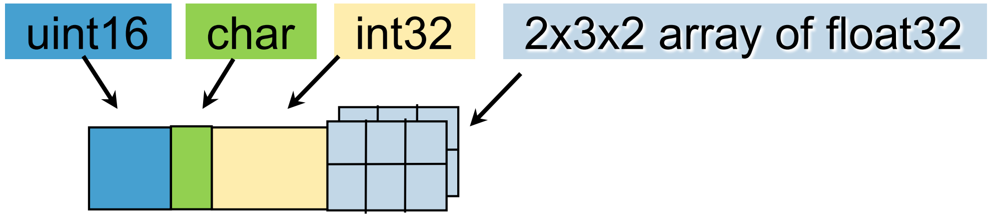
Dataspace: Rank = 2
Dimensions = 3 x 5

Note that this is declared in C as:
“array[5][3]”
and as “array(3)(5)”
in FORTRAN

HDF5 Dataset with Compound Datatype



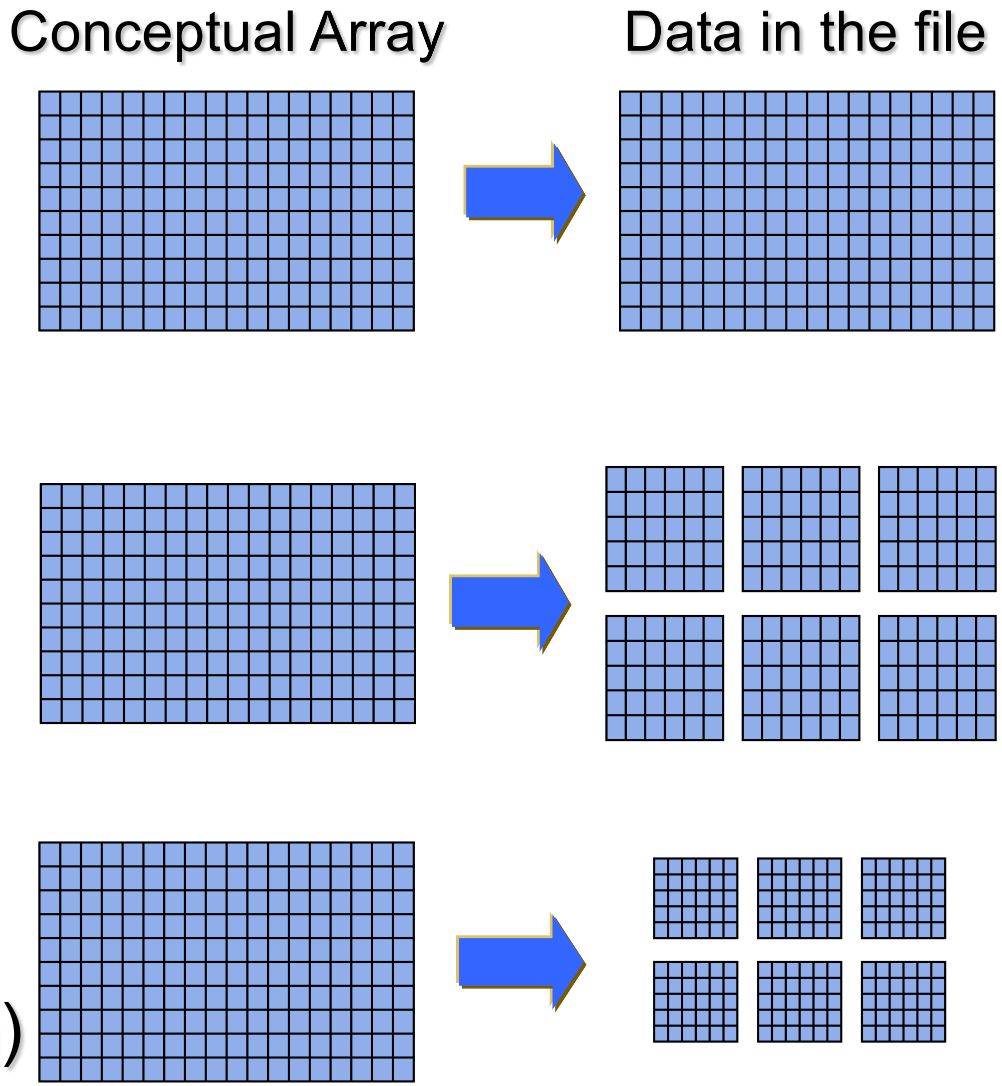
Datatype:



Dataspace: Rank = 2

Dimensions = 3 x 5

Dataset Layout: How are data elements stored?



Contiguous (default)

Data elements stored physically adjacent to each other

Chunked

Better access time for subsets; extendible; can have filters (e.g. compression)

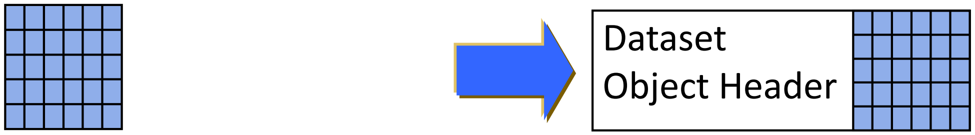
Chunked w/Filters (compression)

Improves storage efficiency, transmission speed, some CPU cost to perform filter

Dataset Layout: How are data elements stored?

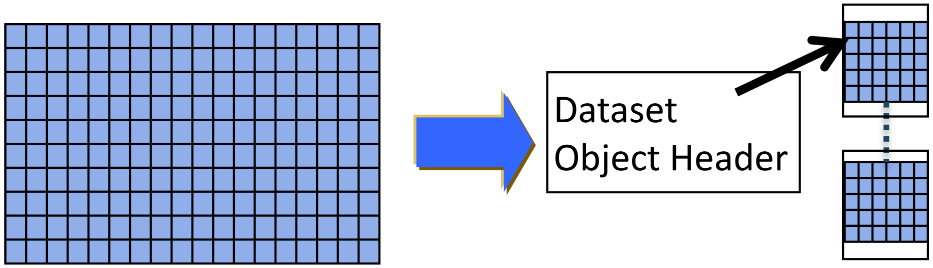
Conceptual Array | Data in the file

Compact



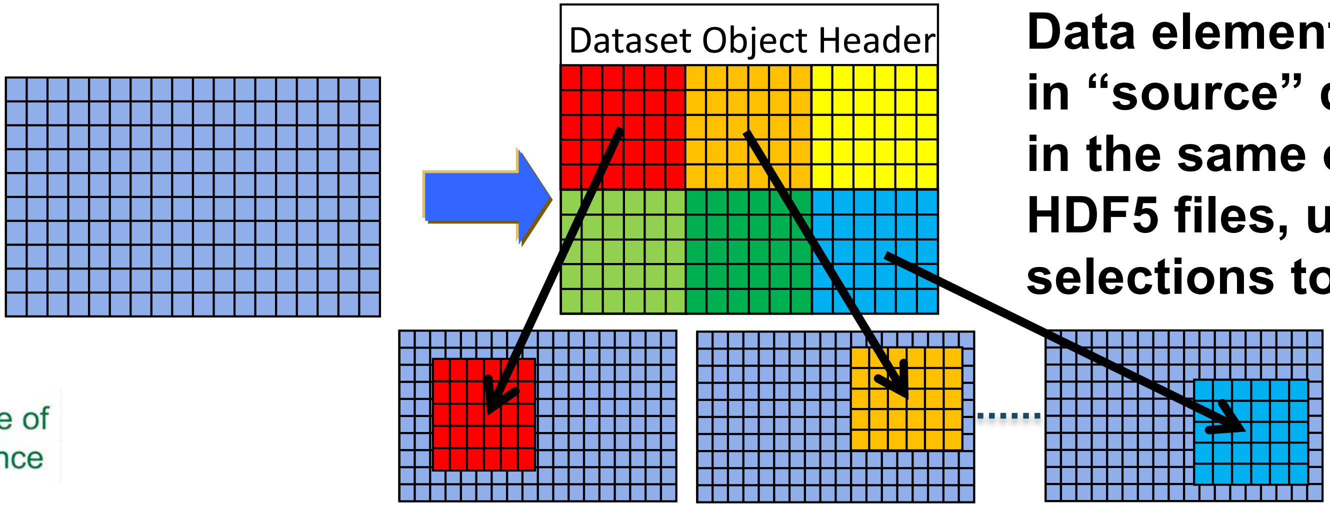
Data elements stored directly within object's metadata

External



Data elements stored outside the HDF5 file, possibly in another file format and / or multiple files

Virtual

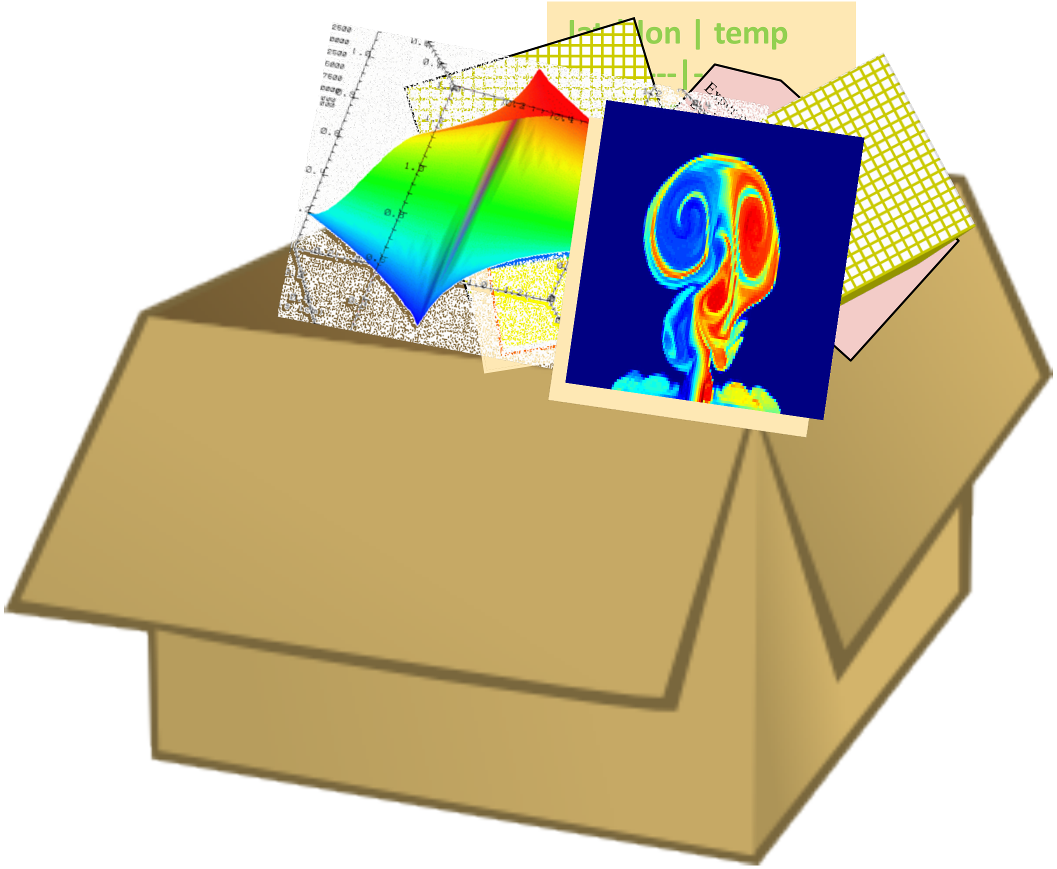


Data elements stored in "source" datasets, in the same or other HDF5 files, using selections to map

HDF5 Attributes

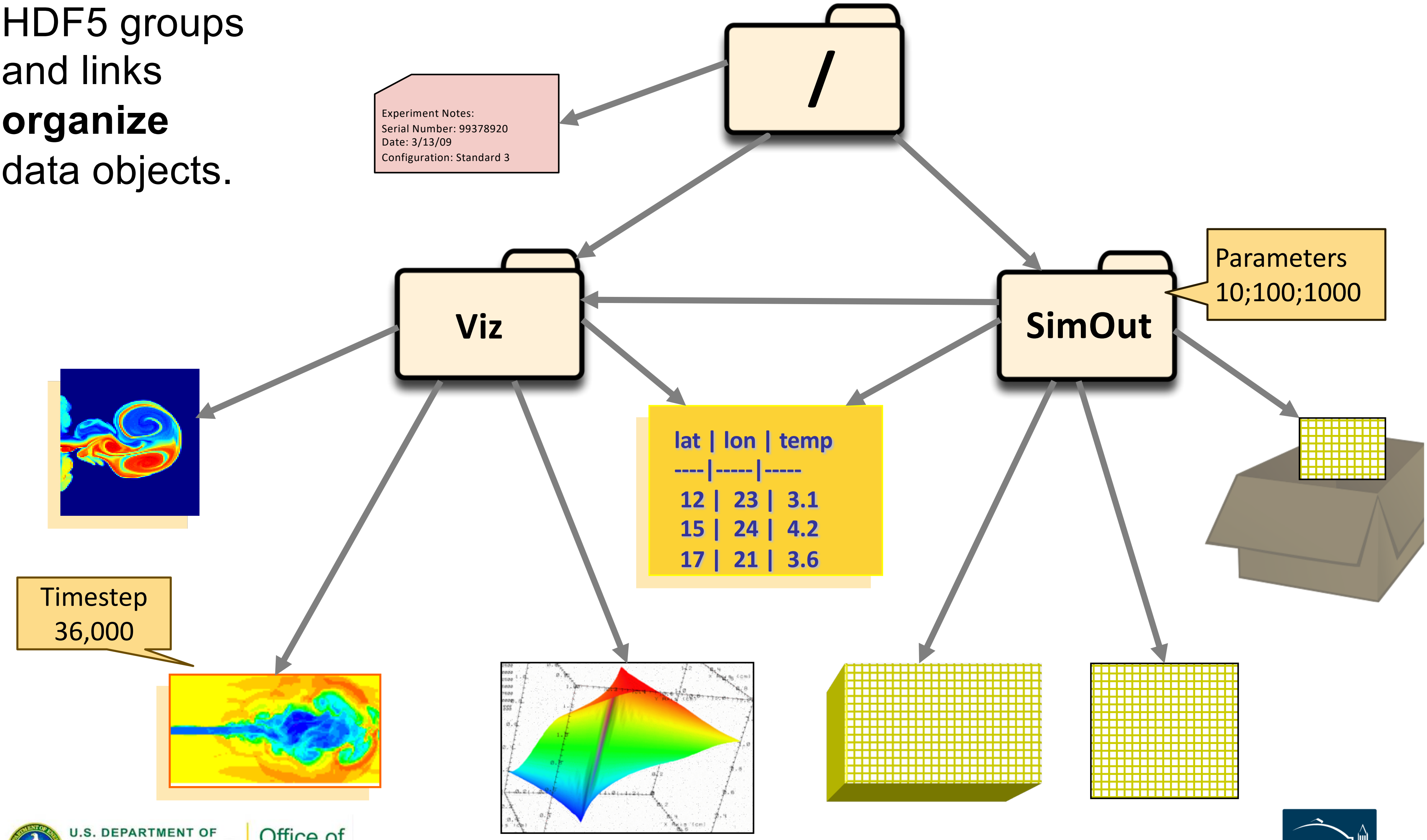
- **Attributes “decorate” HDF5 objects**
- **Typically contain *user* metadata**
- **Similar to “key-value pairs”:**
  - Have a unique name (for that object) and a value
- **Analogous to a dataset**
  - “Value” is an array described by a datatype and a dataspace
  - However, attributes do not support partial I/O operations; nor can they be compressed or extended

HDF5 Groups and Links



HDF5 Groups and Links

HDF5 groups and links **organize** data objects.



Experiment Notes:
Serial Number: 99378920
Date: 3/13/09
Configuration: Standard 3

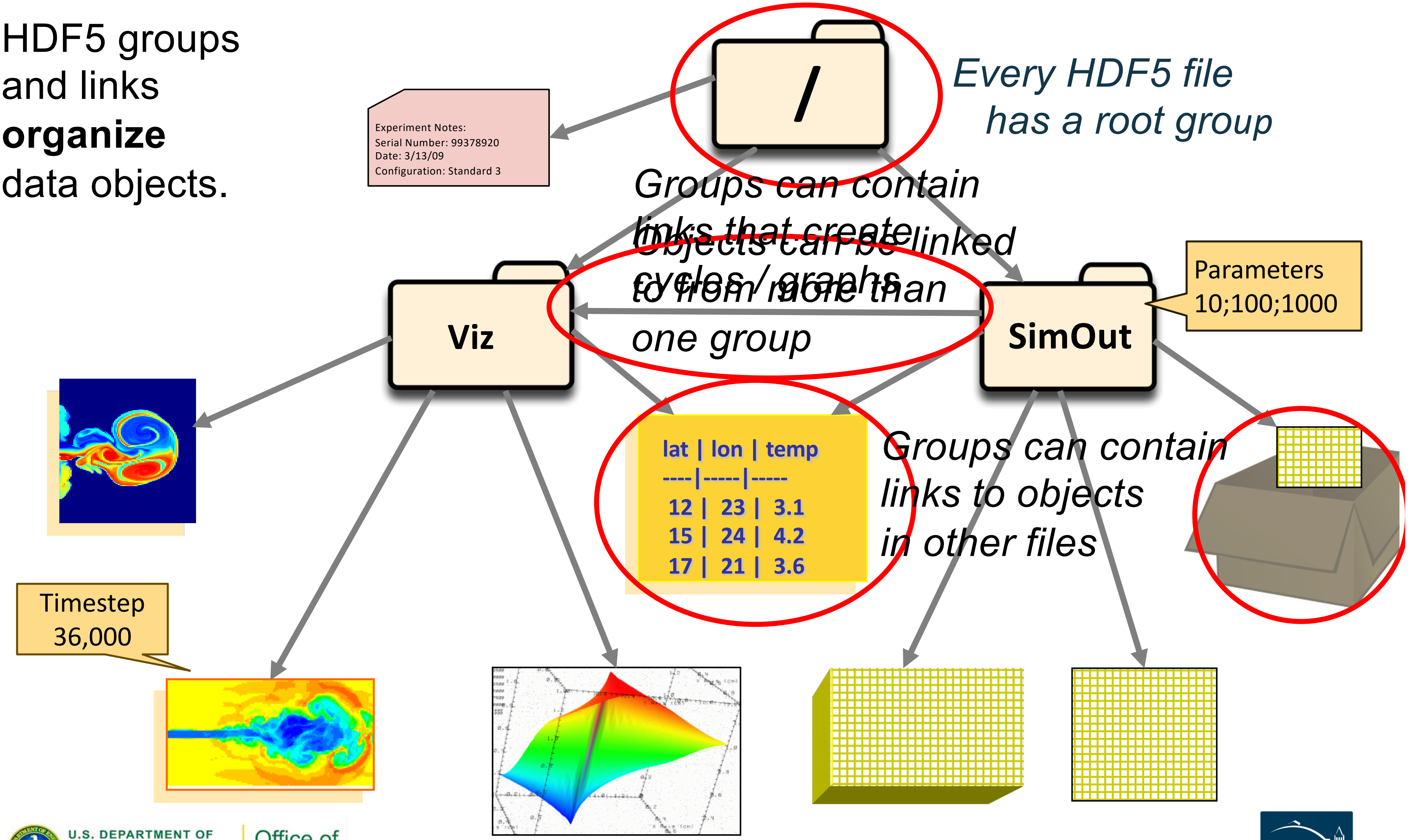
Parameters
10;100;1000

Timestep
36,000

```
lat | lon | temp
----|-----|-----
 12 |  23 |  3.1
 15 |  24 |  4.2
 17 |  21 |  3.6
```

HDF5 Groups and Links

HDF5 groups and links **organize** data objects.



Experiment Notes:
Serial Number: 99378920
Date: 3/13/09
Configuration: Standard 3

Every HDF5 file has a root group

Groups can contain links that create cycles / graphs

Objects can be linked to from more than one group

Groups can contain links to objects in other files

Parameters 10;100;1000

Timestep 36,000

lat	lon	temp
12	23	3.1
15	24	4.2
17	21	3.6

HDF5 SOFTWARE

HDF5 Download Info

Home page:

- https://www.hdfgroup.org/solutions/hdf5

Releases:

- Latest: 1.12.0, with 1.12.1 coming in Fall 2020
- Also supported: 1.10.6 and 1.8.21

Source Distribution:

- https://github.com/HDFGroup/hdf5
- Includes optional language wrappers: C++, FORTRAN, and Java
  - Python available: https://www.h5py.org
- Includes optional High-Level APIs
- Also command-line utilities (h5dump, h5repack, h5diff, …) and compile scripts

Pre-built binaries:

- https://www.hdfgroup.org/downloads/hdf5
- When possible, includes C, C++, FORTRAN , and High Level libraries.
  - Check ./lib/libhdf5.settings file for installed options
- Built with and require the SZIP and ZLIB external libraries

Useful HDF5 Tools For New Users

`h5dump`:

Tool to "dump" or display contents of HDF5 files

`h5cc, h5c++, h5fc`:

Scripts to compile applications (similar to "mpicc")

HDFView:

Java browser to view HDF5 files:

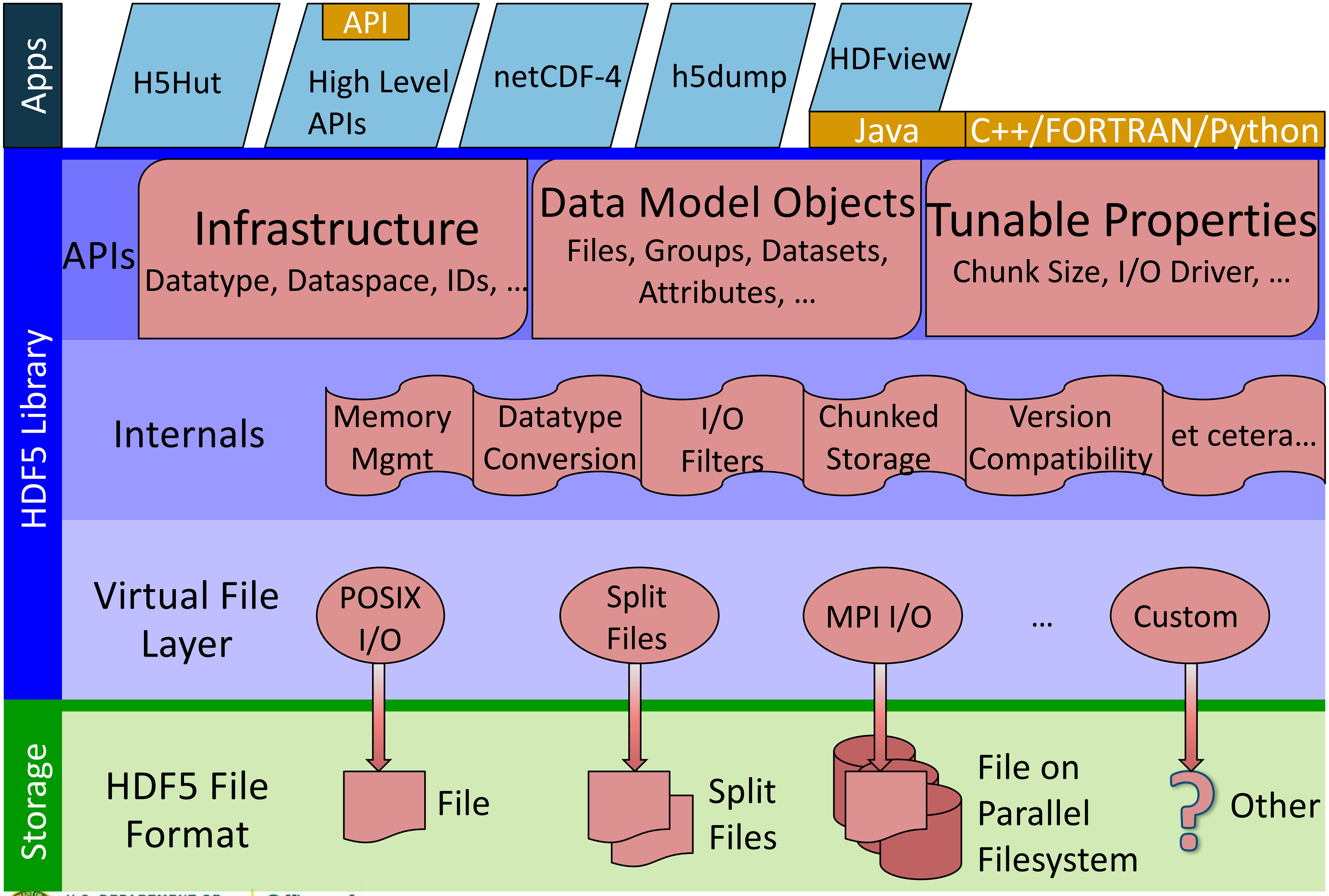
https://support.hdfgroup.org/products/java/hdfview/

HDF5 Examples in C, C++, FORTRAN, Java, Python, Matlab:

https://portal.hdfgroup.org/display/HDF5/HDF5+Examples

PROGRAMMING MODEL AND API

HDF5 Software Layers & Storage



U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

The HDF5 API

- **C, C++, FORTRAN, Java, and .NET bindings**
- **IDL, MATLAB, Python (H5Py, *PyTables*), …**
- **C routines begin with prefix: H5?**

? corresponds to the type of object the function acts on

Example Functions:

H5D:	**D**ataset interface	*e.g.,* **H5Dread**
H5F:	**F**ile interface	*e.g.,* **H5Fopen**
H5S:	data**S**pace interface	*e.g.,* **H5Sclose**

The HDF5 API

- **For flexibility, the API is extensive**
  - 600+ functions!

Victorinox
Swiss Army
Cybertool 34

- **This can be daunting… but there is hope**
  - A few functions can do a lot!
  - Start simple
  - Incrementally build up knowledge and code as more features are needed

General Programming Paradigm

- **Typical for C:**
  - Object is opened or created
  - Object is accessed, possibly many times
  - Object is closed

- **Properties of object or operation are <u>optionally</u> defined:**
  - Creation properties (e.g., use chunking storage)
  - Access properties

Core API Functions

H5Fcreate (H5Fopen)	***create (open) File***
H5Screate_simple	***create dataSpace***
H5Dcreate (H5Dopen)	***create (open) Dataset***
H5Dread / H5Dwrite	***access Dataset***
H5Dclose	***close Dataset***
H5Sclose	***close dataSpace***
H5Fclose	***close File***

Other Common Functions

DataSpaces:	H5Sselect_hyperslab (Partial I/O) H5Sselect_elements (Partial I/O) H5Dget_space
DataTypes:	H5Tcreate, H5Tcommit, H5Tclose H5Dget_type, H5Tequal, H5Tget_native_type
Groups:	H5Gcreate / H5Gopen, H5Gclose
Attributes:	H5Acreate / H5Aopen_name H5Aread, H5Awrite, H5Aclose
Property Lists:	H5Pcreate H5Pset_chunk, H5Pset_deflate H5Pset_fapl_mpio, H5Pset_dxpl_mpio H5Pclose
Other API prefixes:	H5**E** – Errors; H5**I** – IDs; H5**L** – Links; H5**O** – Objects; (and other specialty ones)

PARALLEL HDF5

Terminology

- **“Data” / “Raw Data”**
  - “problem-size” data, e.g., large arrays
- **“Metadata” – is an overloaded term**
- **In this presentation: Metadata “=” HDF5 metadata**
  - For each piece of application metadata, there may be many associated pieces of HDF5 metadata
  - There are also other sources of HDF5 metadata
    - Chunk indices, heaps to store group links and indices to look them up, object headers, etc.

Why Parallel HDF5?

- **Take advantage of high-performance parallel I/O while reducing complexity**
  - Use a high-level I/O layer instead of POSIX or MPI-IO
  - Use only a single (or a few) shared files
    - *“Friends don’t let friends use file-per-process!”* ☺
- **Productivity, Performance, Portability, and Ecosystem**
  - Reduce amount of application code to maintain
  - Rely on HDF5 to optimize for underlying storage system
  - Let HDF5 bear burden of backward / forward compatibility
  - Take advantage of the vast HDF5 ecosystem

What We’ll Cover

- Parallel vs. Serial HDF5
- Implementation Layers
- HDF5 files in a parallel file system
- Parallel HDF5 I/O modes: collective vs. independent
- Data and Metadata I/O

What We Won’t Cover

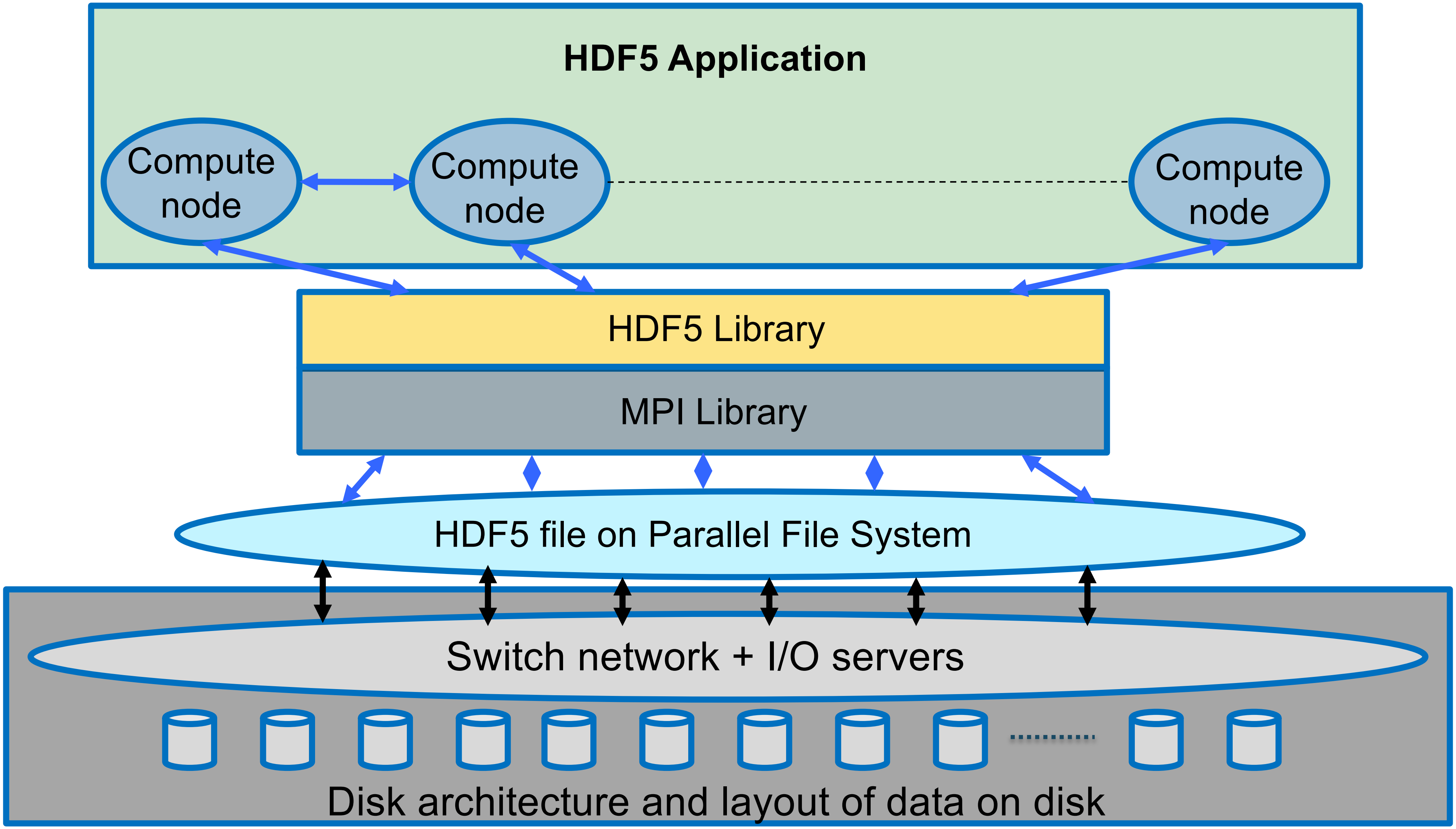
- Consistency semantics
- Virtual Object Layer (VOL)
- Single Writer / Multiple-Reader (SWMR)
- Virtual Datasets (VDS)
- Asynchronous I/O
- Independent Metadata Modification
- …

Contact me on Slack about these after the presentation!

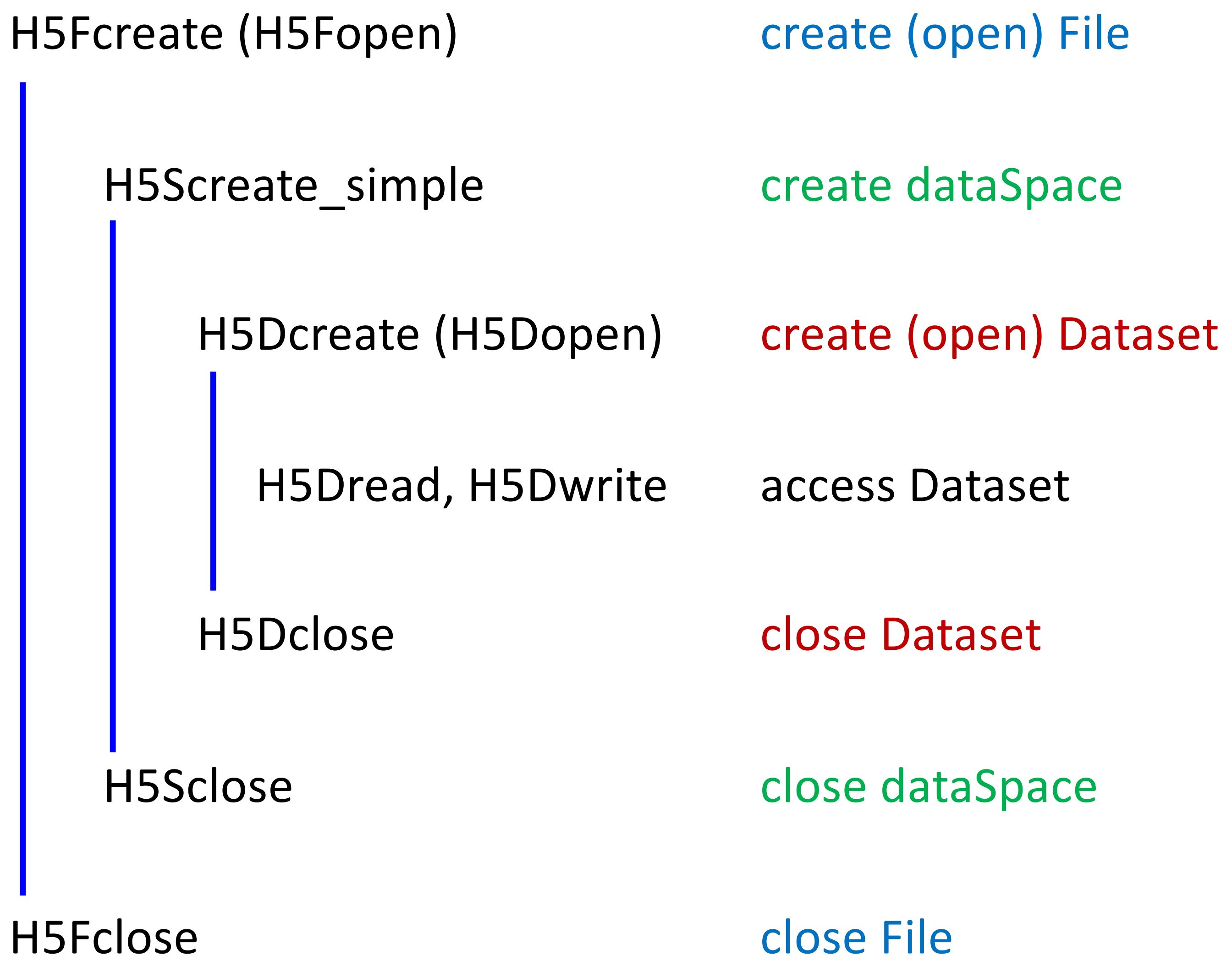
[MPI-] Parallel vs. Serial HDF5

- Parallel HDF5 allows multiple processes in an MPI application to perform I/O to a single HDF5 file
- Uses a standard parallel I/O interface: MPI-IO
  - Portable to different platforms
- Parallel HDF5 files are HDF5 files, conforming to the HDF5 File Format Specification
- The "Parallel HDF5" API consists of:
  - The standard HDF5 API
  - A few extra properties and calls
  - A parallel "etiquette"

PHDF5 Implementation Layers


HDF5 Application
Compute node
Compute node
Compute node
HDF5 Library
MPI Library
HDF5 file on Parallel File System
Switch network + I/O servers
Disk architecture and layout of data on disk

Standard HDF5 “Skeleton”



H5Fcreate (H5Fopen)	create (open) File
H5Screate_simple	create dataSpace
H5Dcreate (H5Dopen)	create (open) Dataset
H5Dread, H5Dwrite	access Dataset
H5Dclose	close Dataset
H5Sclose	close dataSpace
H5Fclose	close File

Example of a Parallel HDF5 C Program

Starting with a simple serial HDF5 program:

```
file_id = H5Fcreate(FNAME, …, H5P_DEFAULT);
space_id = H5Screate_simple(…);
dset_id = H5Dcreate(file_id, DNAME, H5T_NATIVE_INT,
                    space_id, …);


status = H5Dwrite(dset_id, H5T_NATIVE_INT, …, H5P_DEFAULT, …);
…
```

Example of a Parallel HDF5 C Program

A parallel HDF5 program has a few extra calls:

```
MPI_Init(&argc, &argv);

fapl_id = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fapl_mpio(fapl_id, comm, info);
file_id = H5Fcreate(FNAME, …, fapl_id);
space_id = H5Screate_simple(…);
dset_id = H5Dcreate(file_id, DNAME, H5T_NATIVE_INT,
                    space_id, …);
dxpl_id = H5Pcreate(H5P_DATASET_XFER);
H5Pset_dxpl_mpio(xf_id, H5FD_MPIO_COLLECTIVE);
status = H5Dwrite(dset_id, H5T_NATIVE_INT, …, dxpl_id, …);
…
MPI_Finalize();
```

Parallel HDF5 “Etiquette”

- **Parallel HDF5 opens a shared file with an MPI communicator**
  - Returns a file ID (as usual)
  - All future access to the file via that file ID (as usual)
  - However, this file ID can be used to open datasets and then perform collective data I/O operations
- **Different files can be opened via different communicators**

- **All processes must participate in collective HDF5 APIs**
- **All HDF5 APIs that modify the HDF5 *namespace* and *structural metadata* are collective!**
  - File ops., group structure, dataset dimensions, object life-cycle, etc.
- **Debugging metadata hangs:**
  - Can “bisect” with H5Fflush in source code
  - Or, can set H5_COLL_API_SANITY_CHECK environment variable:
    - “setenv H5_COLL_API_SANITY_CHECK 1”

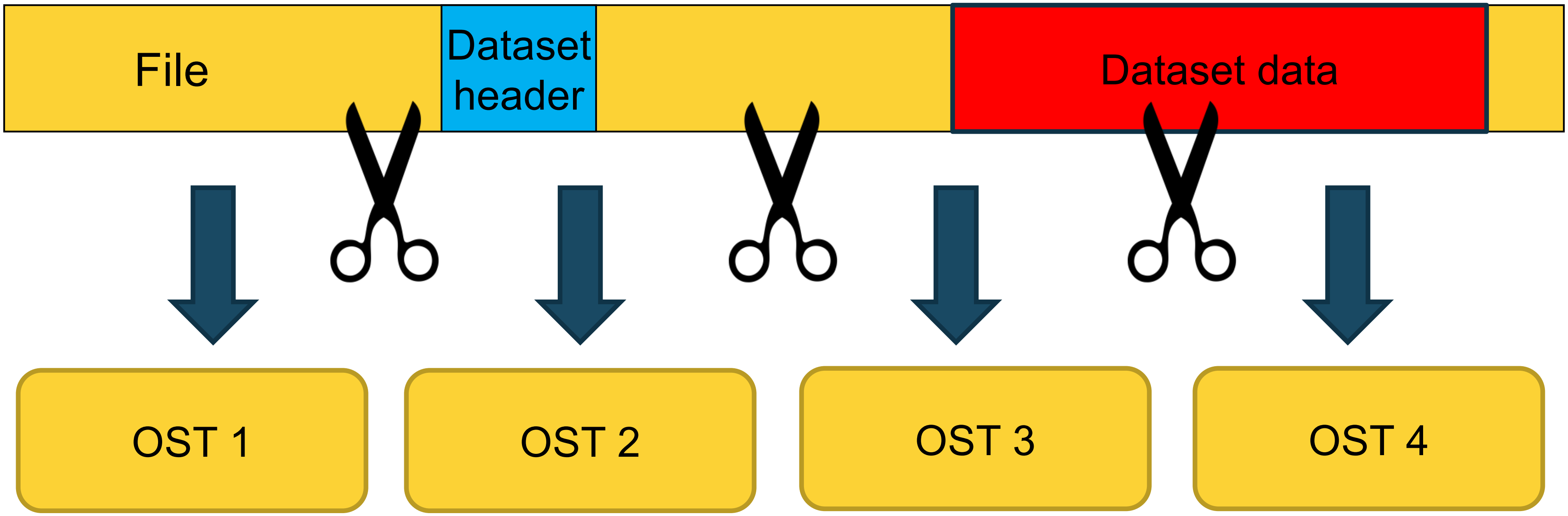
https://support.hdfgroup.org/HDF5/doc/RM/CollectiveCalls.html

Parallel HDF5 Tutorial Examples

- **For more examples how to write different data patterns see:**

https://portal.hdfgroup.org/display/HDF5/Introduction+to+Parallel+HDF5

In a Parallel File System

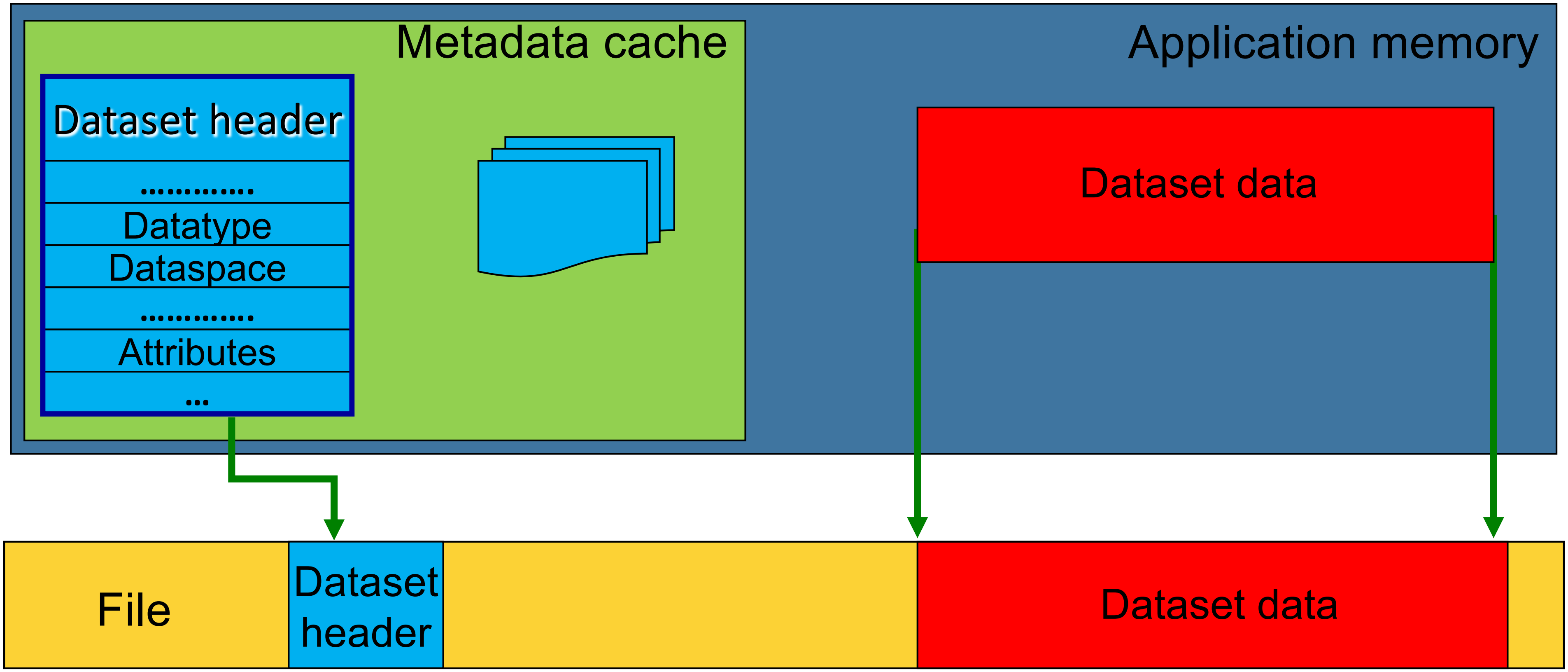


The file is striped over multiple “disks” (e.g. Lustre OSTs) depending on the stripe size and stripe count with which the file was created.

And it gets worse before it gets better…

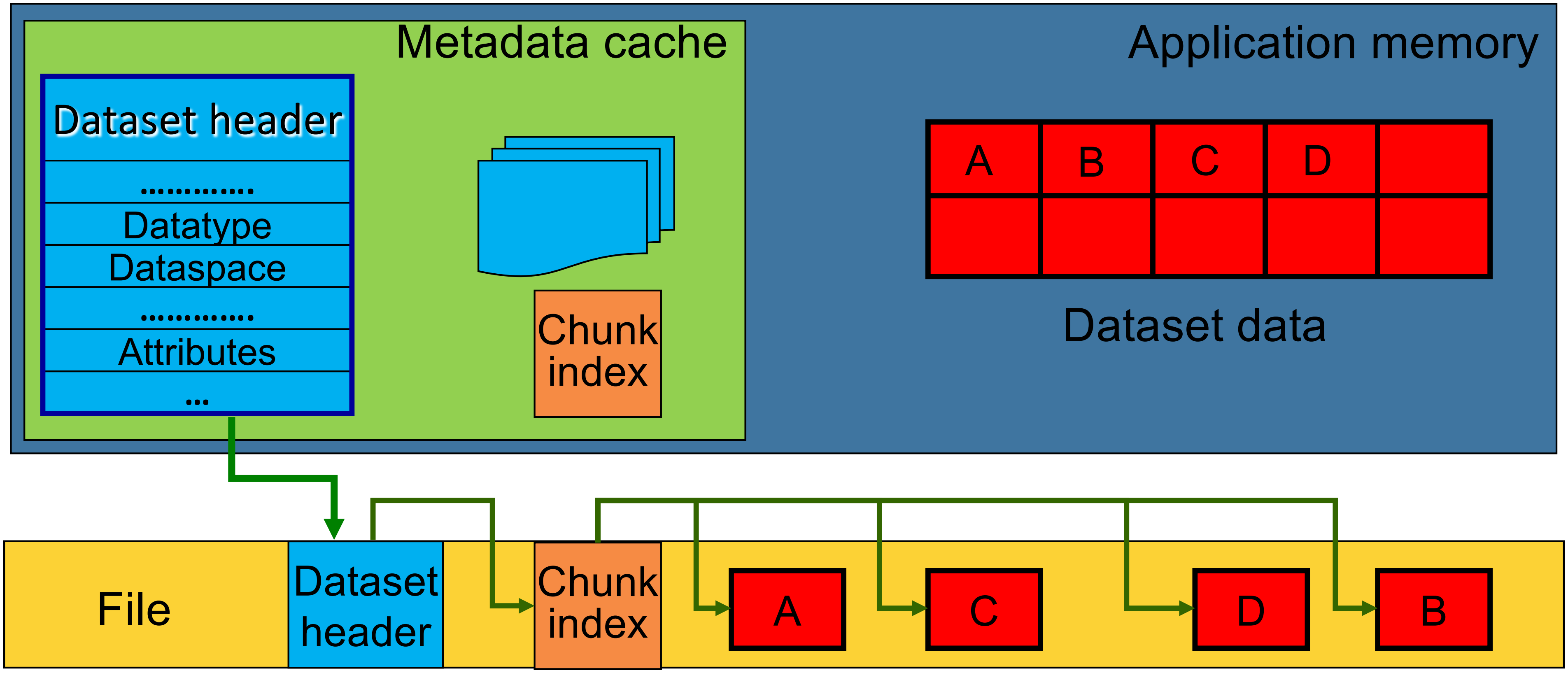
HDF5 - Contiguous Storage

- **HDF5 Object header, separate from dataset data**
- **Data stored in one contiguous block in HDF5 file**

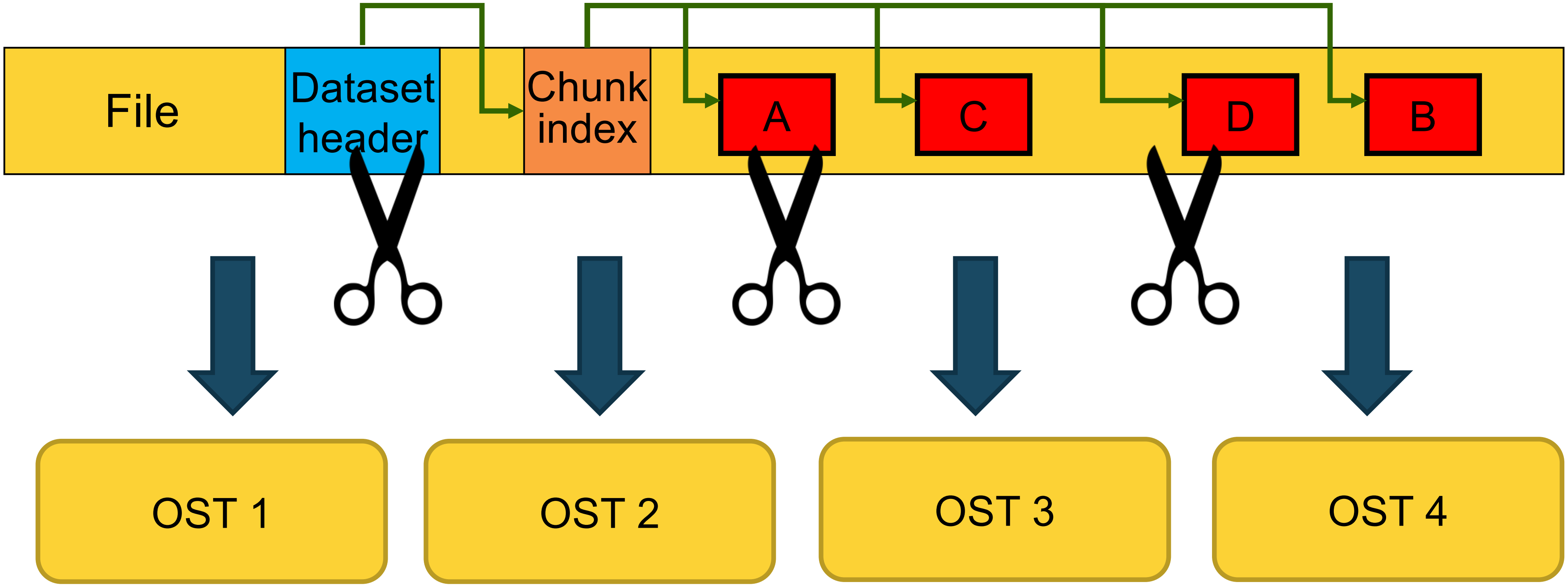


HDF5 - Chunked Storage

- **Dataset data is divided into equal-sized blocks (chunks)**
- **Each chunk is stored separately in the HDF5 file, located by a “chunk index”**



HDF5 In a Parallel File System



The file is striped over multiple OSTs depending on the stripe size and stripe count with which the file was created.

Collective vs. Independent I/O

- **Collective I/O <u>attempts</u> to combine multiple smaller independent I/O ops into fewer larger ops.**
  - Neither mode is preferable *a priori*
- **MPI definition of collective calls:**
  - All processes of the communicator must participate in calls in the same order:

Process 1	Process 2	
call A(); ➔ call B();	call A(); ➔ call B();	**right**
call A(); ➔ call B();	call B(); ➔ call A();	**wrong**

  - Independent calls are not collective ☺
  - Collective calls are not necessarily synchronous, nor must they require communication
    - It could be that only internal state for the communicator changes

Data and Metadata I/O in Parallel HDF5

Data

- **“Problem-sized”**
- **I/O can be independent or collective**

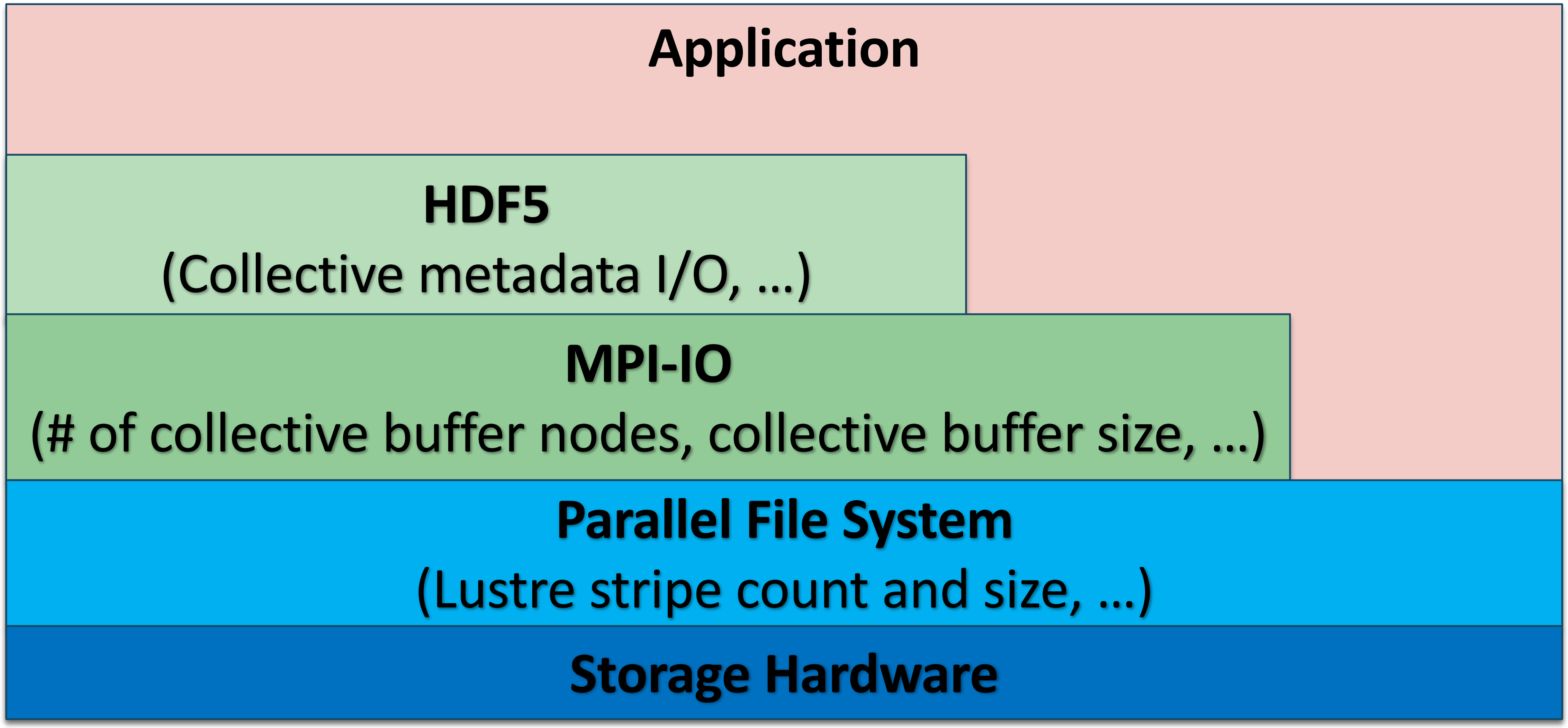
- **Improvement targets:**
  - Alignment
  - Avoid datatype conversion
  - Reduce I/O frequency
  - Pay attention to layout on disk
    - Different I/O strategies for chunked layout
  - Aggregation and balancing

Metadata

- **“Small”**
- **Reads can be independent or collective**
- **All modifications must be collective**

- **Improvement targets:**
  - User-level metadata / namespace design
  - Use the latest library version, if possible
  - Metadata cache
    - In desperate cases, take control of evictions

Don’t Forget! It’s a Multi-Layer Problem



Application

HDF5
(Collective metadata I/O, …)

MPI-IO
(# of collective buffer nodes, collective buffer size, …)

Parallel File System
(Lustre stripe count and size, …)

Storage Hardware